

Reduced basis methods for parametrized non-linear evolution equations

Martin Drohmann

Institute of Numerical and Applied Mathematics, Münster <http://www.math.uni-muenster.de/num>

26/05/2012

Motivation: Reduced Basis Method

RB Scenario:

- ▶ Parametrized applications relying on **time-critical** or many **repeated** simulations

Goals:

- ▶ **Offline**-/Online decomposition
- ▶ Efficient reduced simulations
- ▶ A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Motivation: Reduced Basis Method

RB Scenario:

- ▶ Parametrized applications relying on **time-critical** or many **repeated** simulations

Goals:

- ▶ **Offline**-/Online decomposition
- ▶ Efficient reduced simulations
- ▶ A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Motivation: Reduced Basis Method

RB Scenario:

- ▶ Parametrized applications relying on **time-critical** or many **repeated** simulations

Goals:

- ▶ Offline-/Online decomposition
- ▶ Efficient **reduced** simulations
- ▶ A posteriori error control

References: [Patera&Rozza, 2006], [Haasdonk et al., 2008]

Idea of the Reduced Basis Method

Parametrized PDE

Find

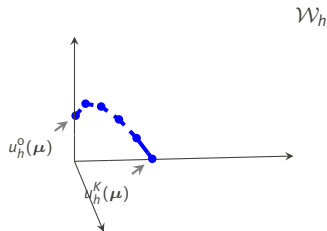
$$u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega), \text{ s.t.}$$

$$u(0) = u_0$$

$$\partial_t u(t) - \mathcal{L} \quad [u(t)] = 0$$

plus
conditions.

boundary



Idea of the Reduced Basis Method

Parametrized PDE

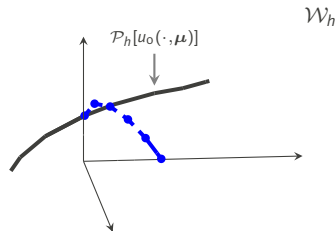
Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L} \quad [u(t)] = 0$$

plus
conditions.

boundary



Idea of the Reduced Basis Method

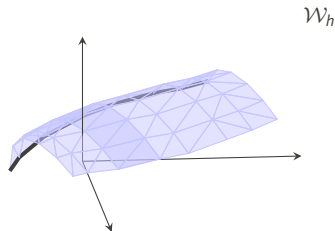
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

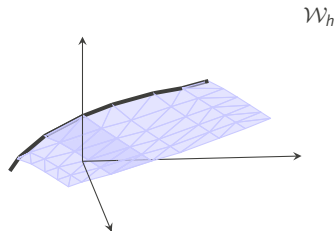
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

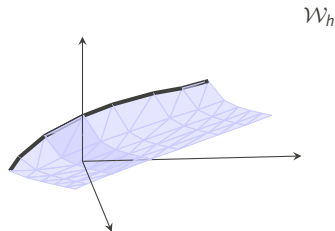
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

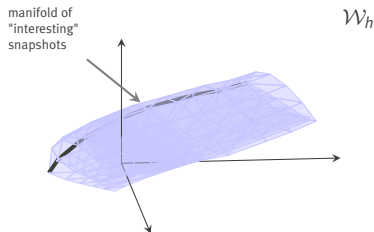
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

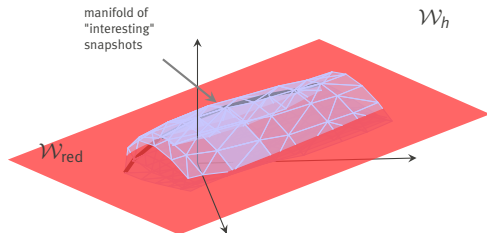
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

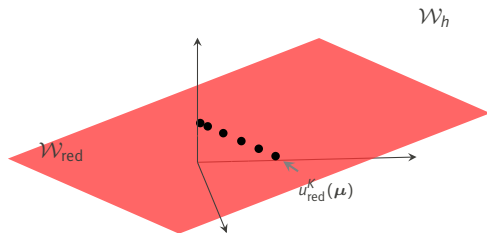
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Idea of the Reduced Basis Method

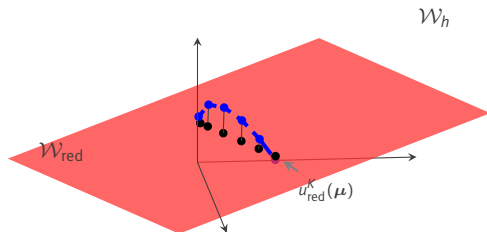
Parametrized PDE

Find, for $\mu \in \mathcal{P} \subset \mathbb{R}^p$,
 $u : [0, T_{\max}] \rightarrow \mathcal{W} \subset L^2(\Omega)$, s.t.

$$u(0) = u_0(\mu)$$

$$\partial_t u(t) - \mathcal{L}(\mu)[u(t)] = 0$$

plus (parameter dependent) boundary conditions.



Example: FV scheme for Burgers-Equation

Burgers Equation

$$\partial_t u - \nabla \mathbf{v} u^{\mu_1} = 0 \quad (1)$$

e.g. discretized by finite volume method with Engquist–Osher flux.

Example: FV scheme for Burgers-Equation

Burgers Equation

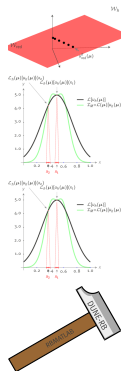
$$\partial_t u - \nabla \mathbf{v} u^{\mu_1} = 0 \quad (1)$$

e.g. discretized by finite volume method with Engquist–Osher flux.

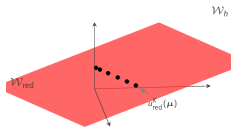
- ▶ Parameter vector $\boldsymbol{\mu} := (\mu_1) \in [1, 2]$.
- ▶ smooth initial data
- ▶ rectangular 120×60 grid with $K = 100$ time steps.

Questions

1. How can we compute **efficient** reduced simulations?
2. How can we compute efficient reduced simulations for **non-linear** problems?
3. How can we **generate** the reduced basis space?
4. How can we apply the RB method to our favorite problems?



1. How can we compute **efficient** reduced simulations?



Reduced basis scheme (Haasdonk et al., 1997)

Reduced simulation (linear with affine parameter dependence)

For $\mu \in \mathcal{P}$ find $\{u_h^k(\mu)\}_{k=0}^K \subset \mathcal{W}_h$, such that

$$u_h^0(\mu) := \mathcal{P}_h[u_0(\mu)]$$

$$(\text{Id} - \Delta t \mathcal{L}_{h,I}(\mu)) [u_h^{k+1}(\mu)] = (\text{Id} + \Delta t \mathcal{L}_{h,E}(\mu)) [u_h^k(\mu)].$$

Reduced basis scheme (Haasdonk et al., 1997)

Reduced simulation (linear with affine parameter dependence)

For $\mu \in \mathcal{P}$ find $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$\begin{aligned} u_h^0(\mu) &:= \mathcal{P}_h[u_0(\mu)] \\ (\text{Id} - \Delta t \mathcal{L}_{h,I}(\mu)) [u_h^{k+1}(\mu)] &= (\text{Id} + \Delta t \mathcal{L}_{h,E}(\mu)) [u_h^k(\mu)]. \end{aligned}$$

Reduced basis scheme (Haasdonk et al., 1997)

Reduced simulation (linear with affine parameter dependence)

For $\mu \in \mathcal{P}$ find $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$u_{\text{red}}^0(\mu) := \mathcal{P}_{\text{red}}[u_0(\mu)]$$
$$(\text{Id} - \Delta t \mathcal{L}_{h,I}(\mu)) [u_h^{k+1}(\mu)] = (\text{Id} + \Delta t \mathcal{L}_{h,E}(\mu)) [u_h^k(\mu)].$$

Size: $N \times N$
Structure:
Dense

Size:
 $N \times 1$

=

Size: $N \times N$
Structure:
Sparse

Size:
 $N \times 1$

Reduced basis scheme (Haasdonk et al., 1997)

Reduced simulation (linear with affine parameter dependence)

For $\mu \in \mathcal{P}$ find $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$u_{\text{red}}^0(\mu) := \mathcal{P}_{\text{red}}[u_0(\mu)]$$
$$(\text{Id} - \Delta t \mathcal{L}_{h,I}(\mu)) [u_{\text{red}}^{k+1}(\mu)] = (\text{Id} + \Delta t \mathcal{L}_{h,E}(\mu)) [u_{\text{red}}^k(\mu)].$$

Size: $N \times N$
Structure:
Dense

Size:
 $N \times 1$

=

Size: $N \times N$
Structure:
Sparse

Size:
 $N \times 1$

Reduced basis scheme (Haasdonk et al., 1997)

Reduced simulation (linear with affine parameter dependence)

For $\mu \in \mathcal{P}$ find $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$u_{\text{red}}^0(\mu) := \mathcal{P}_{\text{red}}[u_0(\mu)]$$
$$(\text{Id} - \Delta t \mathcal{L}_{\text{red}, I}(\mu)) [u_{\text{red}}^{k+1}(\mu)] = (\text{Id} + \Delta t \mathcal{L}_{\text{red}, E}(\mu)) [u_{\text{red}}^k(\mu)].$$

Size: $N \times N$
Structure:
Dense

Size:
 $N \times 1$

=

Size: $N \times N$
Structure:
Sparse

Size:
 $N \times 1$

Offline-/Online decomposition

Affine parameter dependence

Assume $\mathcal{L}_{h,I/E}$ can be written as

$$\mathcal{L}_{h,I/E}(\mu)[u_h] = \sum_{q=1}^{Q_{I/E}} \sigma_{I/E}^q(\mu) \mathcal{L}_{h,I/E}^q[u_h].$$

$$\sum_{q=1}^{Q_{I/E}}$$



Size: $H \times H$
Structure: Sparse

Offline-/Online decomposition

Affine parameter dependence

Assume $\mathcal{L}_{h,I/E}$ can be written as

$$\mathcal{L}_{h,I/E}(\mu)[u_h] = \sum_{q=1}^{Q_{I/E}} \sigma_{I/E}^q(\mu) \mathcal{L}_{h,I/E}^q[u_h].$$

$$\sum_{q=1}^{Q_{I/E}}$$



Size: $H \times H$
Structure: Sparse

Offline-/Online decomposition

Affine parameter dependence

Assume $\mathcal{L}_{h,I/E}$ can be written as

$$\mathcal{L}_{h,I/E}(\mu)[u_h] = \sum_{q=1}^{Q_{I/E}} \sigma_{I/E}^q(\mu) \mathcal{L}_{h,I/E}^q[u_h].$$

Reduced operators are: $\mathcal{L}_{\text{red},I/E}(\mu) = \mathcal{P}_{\text{red}}[\mathcal{L}_{h,I/E}(\mu)]$

$$\sum_{q=1}^{Q_{I/E}}$$



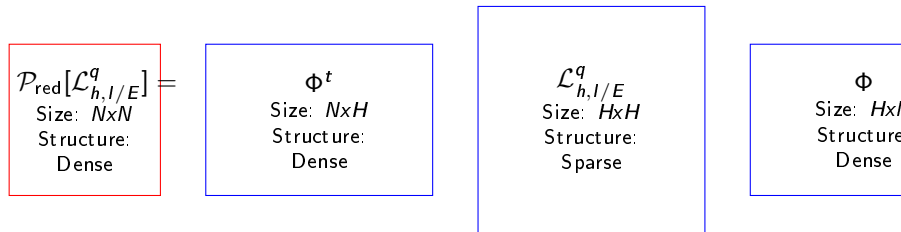
Size: $H \times H$
Structure: Sparse

Offline-/Online decomposition

Assume RB space spanned by N basis functions $\Phi := \{\varphi_i\}_{i=1}^N$.

Offline matrices

Reduction of parameter independent components: $\mathcal{P}_{\text{red}} \left[\mathcal{L}_{h,I/E}^q \right]$



Offline-/Online decomposition



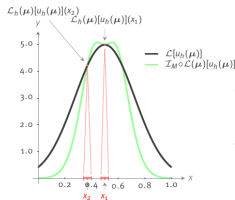
$$\mathcal{L}_{\text{red}, I/E}(\mu) = \sum_{q=1}^{Q_{I/E}} \sigma_{I/E}^q(\mu) \mathcal{P}_{\text{red}}[\mathcal{L}_{h, I/E}^q]$$

$$\mathcal{L}_{\text{red}, I/E}(\mu) = \sum_{q=1}^{Q_{I/E}}$$



Size: $N \times N$
Structure:
Dense

2. How can we compute efficient reduced simulations for **non-linear** problems?



Reduced basis scheme (DHO, 2012)

Reduced simulation (implicit/explicit with Newton scheme)

For $\mu \in \mathcal{P}$ find $\{u_{\text{red}}^k(\mu)\}_{k=0}^K \subset \mathcal{W}_{\text{red}} \subset \mathcal{W}_h$, such that

$$u_{\text{red}}^0 := \mathcal{P}_{\text{red}}[u_0(\mu)], \quad u_{\text{red}}^{k+1} := u_{\text{red}}^{k+1, \nu_{\max}(k)}$$

with Newton iteration

$$u_{\text{red}}^{k+1,0} := u_{\text{red}}^k, \quad u_{\text{red}}^{k+1,\nu+1} := u_{\text{red}}^{k+1,\nu} + \delta_{\text{red}}^{k+1,\nu+1},$$
$$\left(\text{Id} + \Delta t \mathbf{D} \mathcal{L}_{\text{red}, I?} \Big|_{u_{\text{red}}^{k+1,\nu}} \right) [\delta_{\text{red}}^{k+1,\nu+1}] = u_{\text{red}}^k - u_{\text{red}}^{k+1,\nu} - \Delta t \left(\mathcal{L}_{\text{red}, I?} \left[u_{\text{red}}^{k+1,\nu} \right] + \mathcal{L}_{\text{red}, E?} \left[u_{\text{red}}^k \right] \right)$$

What to do with the operators?

Problem

No affine parameter decomposition...

What to do with the operators?

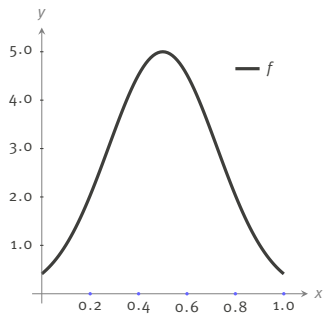
Problem

No affine parameter decomposition...

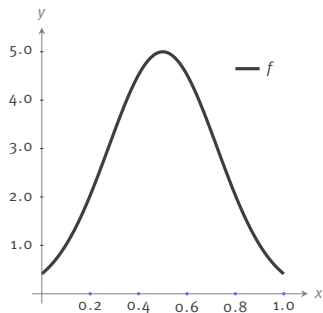
Answer

Empirical operator interpolation!

What is interpolation?

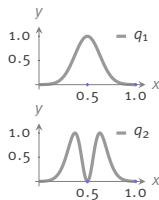


What is interpolation?

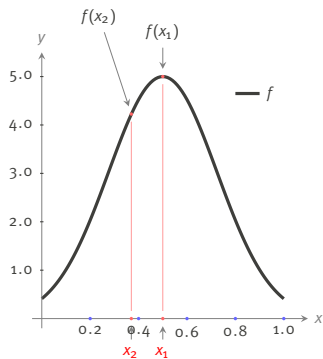


► interpolation space $\mathcal{X}^{\mathcal{I}}$

Base functions:

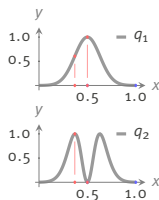


What is interpolation?



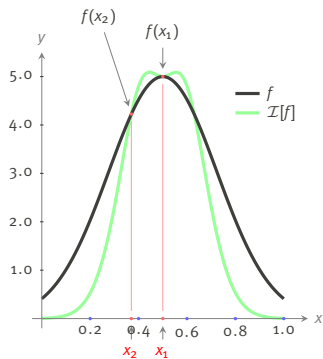
► interpolation space $\mathcal{X}^{\mathcal{I}}$

Base functions:



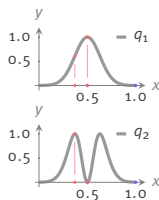
► interpolation nodes $T^{\mathcal{I}}$

What is interpolation?



► interpolation space $\mathcal{X}^{\mathcal{I}}$

Base functions:



► interpolation nodes $T^{\mathcal{I}}$

Example: polynomial interpolation

Polynomial interpolation

- ▶ $\mathcal{X} := L^\infty([0, 1])$
- ▶ $\mathcal{X}^{\mathcal{I}_p} := \Pi_p$ polynomials of degree p .
- ▶ $T^{\mathcal{I}_p}$ e.g. Chebyshev points.

Example: polynomial interpolation

Polynomial interpolation

- ▶ $\mathcal{X} := L^\infty([0, 1])$
- ▶ $\mathcal{X}^{\mathcal{I}_p} := \Pi_p$ polynomials of degree p .
- ▶ $T^{\mathcal{I}_p}$ e.g. Chebyshev points.
- ▶ works only in 1D
- ▶ polynomials not always best selection

Empirical interpolation[Barrault et al, 2004]

Empirical interpolation

- ▶ $\mathcal{X} := \{f(\mu); \mu \in \mathcal{P}\} \subset L^\infty(\Omega)$
- ▶ $\mathcal{X}^{\mathcal{I}_M} := \text{span}\{q_m\}_{m=1}^M$ “empirically” determined.
- ▶ $\mathcal{T}^{\mathcal{I}_M}$ “empirically” determined.

Empirical interpolation[Barrault et al, 2004]

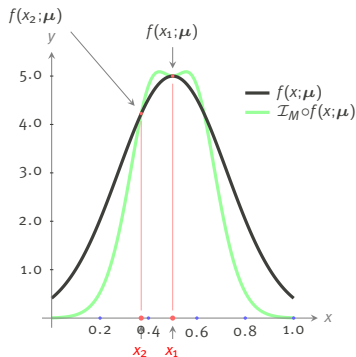
Empirical interpolation

- ▶ $\mathcal{X} := \{f(\mu); \mu \in \mathcal{P}\} \subset L^\infty(\Omega)$
- ▶ $\mathcal{X}^{\mathcal{I}_M} := \text{span}\{q_m\}_{m=1}^M$ “empirically” determined.
- ▶ $\mathcal{T}^{\mathcal{I}_M}$ “empirically” determined.

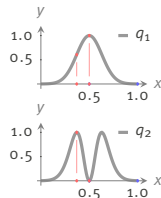
For selection of interpolation nodes and functions: ▶ *El-greedy*

Empirical interpolation[Barrault et al, 2004]

Empirical interpolation for parametrized functions $f(\mu) : \mathbb{R} \rightarrow \mathbb{R}$.



Base functions:

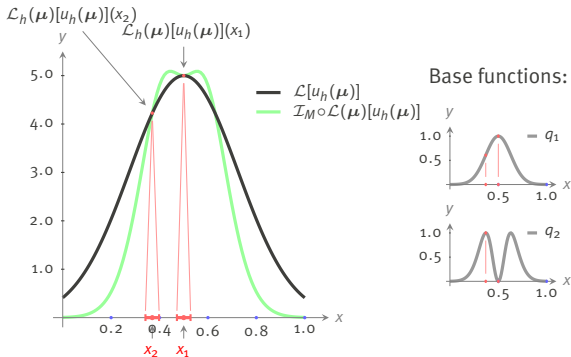


► “magic points” $\{x_m\}_{m=1}^M$

► basis functions $\{q_m\}_{m=1}^M$

Empirical operator interpolation[DHO11]

Empirical interpolation for parametrized discrete operators $\mathcal{L}_h \in \mathcal{W}_h$.

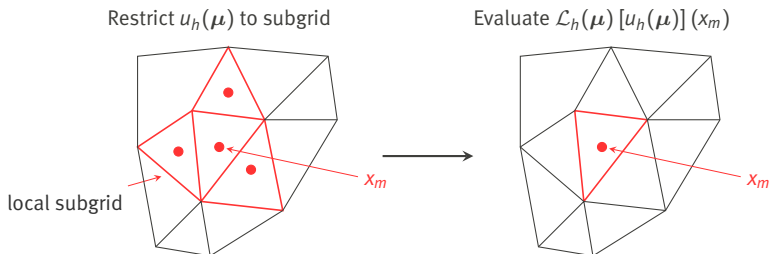


► “magic points” $\{x_m\}_{m=1}^M$

Discrete operators need to have “*H*-independent Dof dependence”.

► basis functions $\{q_m\}_{m=1}^M$

Empirical operator interpolation: Subgrids



Efficient evaluations

The operator evaluations in interpolation points $\mathcal{L}_h(\mu)[\cdot](x_m)$ can be computed efficiently during **online** phase, if

- ▶ the operator has a localized structure (**small stencil**) and
- ▶ the local geometry information is precomputed during **offline** phase.

Empirical interpolation: Fréchet derivative

Define $l_m(\boldsymbol{\mu}) : \mathcal{W}_h \rightarrow \mathbb{R}, u_h \mapsto \mathcal{L}_h(\boldsymbol{\mu})[u_h](x_m)$

Observation

$$\left(\mathbf{D} \left(\mathcal{I}_M [\mathcal{L}_h(\boldsymbol{\mu})] \right) |_{u_h} [v_h] \right) (x_m) = \mathbf{D} l_m(\boldsymbol{\mu})|_{u_h} [v_h]$$

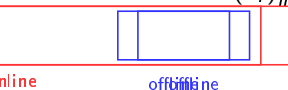
still **efficiently** computable with complexity $\mathcal{O}(M)$.

Offline-/Online Decomposition

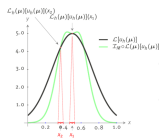
Compute matrices $\mathbf{L}_{I/E}, \mathbf{J}_I \in \mathbb{R}^{N \times M}$ depending on u_{red} and δ_{red} :

$$(\mathbf{L}_{I/E})_{nm} = \sum_{m=1}^M l_m^{I/E}(\mu)[u_{\text{red}}] \int_{\Omega} q_m \varphi_n$$

$$(\mathbf{J}_I)_{nm} = \sum_{m=1}^M \mathbf{D} l_m^{I/E}(\mu)|_{u_{\text{red}}}[\delta_{\text{red}}] \int_{\Omega} q_m \varphi_n$$



3. How can we **generate** the reduced basis space?



Basis Generation methods

Reduce an “intelligent” selection of **snapshots** by

- ▶ Proper orthogonal decomposition,
- ▶ Greedy algorithms or
- ▶ a combination of both.

Basis Generation methods

Reduce an “intelligent” selection of **snapshots** by

- ▶ Proper orthogonal decomposition,
- ▶ Greedy algorithms or
- ▶ a combination of both.

Plus: Selection of “magic points” for empirical operator interpolation.

Basis Generation methods

Reduce an “intelligent” selection of **snapshots** by

- ▶ Proper orthogonal decomposition,
- ▶ Greedy algorithms or
- ▶ a combination of both.

Plus: Selection of “magic points” for empirical operator interpolation.

Challenge: Control of error and basis sizes.

A posteriori error estimator (DHO 2012)

Aim

$$\|u_h^k(\mu) - u_{\text{red}}^k(\mu)\| \leq \eta^k(\mu)$$

Two main contributions:

- ▶ Projection error on \mathcal{W}_{red} (exactly computable!)
- ▶ Empirical interpolation error ($M + M'$ trick)

A posteriori error estimator

Theorem (A posteriori error estimator)

Assumptions:

- ▶ Operators fulfill “Lipschitz” properties:
 - ▶ $\|u - v + \Delta t \mathcal{L}_I[u] - \Delta t \mathcal{L}_I[v]\|_{\mathcal{W}_h} \geq \frac{1}{C_{I,\Delta t}} \|u - v\|_{\mathcal{W}_h}$
 - ▶ $\|u - v - \Delta t \mathcal{L}_E[u] + \Delta t \mathcal{L}_E[v]\|_{\mathcal{W}_h} \leq C_{E,\Delta t} \|u - v\|_{\mathcal{W}_h}$
- ▶ M' -trick: Empirical interpolations exact for larger CRB space $\mathcal{W}_{M+M'}$ and $\mathcal{P}_h[u_0(\mu)] \in \mathcal{W}_{red}$

A posteriori error estimator

Theorem (A posteriori error estimator cont.)

Assumptions:

- ▶ Operators fulfill “Lipschitz” properties:
 - ▶ $\|u - v + \Delta t \mathcal{L}_I[u] - \Delta t \mathcal{L}_I[v]\|_{\mathcal{W}_h} \geq \frac{1}{C_{I,\Delta t}} \|u - v\|_{\mathcal{W}_h}$
 - ▶ $\|u - v - \Delta t \mathcal{L}_E[u] + \Delta t \mathcal{L}_E[v]\|_{\mathcal{W}_h} \leq C_{E,\Delta t} \|u - v\|_{\mathcal{W}_h}$
- ▶ M' -trick: Empirical interpolations exact for larger CRB space $\mathcal{W}_{M+M'}$ and $\mathcal{P}_h[u_0(\mu)] \in \mathcal{W}_{red}$

Then:

$$\|u_{red}^k(\mu) - u_h^k(\mu)\| \leq \eta_{N,M}^k(\mu)$$

with

$$\eta_{N,M}(\mu) := \sum_{i=0}^{k-1} C_{I,\Delta t}^{k-i+1} C_{E,\Delta t}^{k-i} \left(\|R_{I+E,M}^{k+1}(\mu)\| + \|R^{k+1}(\mu)\| \right)$$

A posteriori error estimator

Theorem (A posteriori error estimator)

Then:

$$\left\| u_{\text{red}}^k(\mu) - u_h^k(\mu) \right\| \leq \eta_{N,M}^k(\mu)$$

with

$$\eta_{N,M}(\mu) := \sum_{i=0}^{k-1} C_{I,\Delta t}^{k-i+1} C_{E,\Delta t}^{k-i} \left(\left\| R_{I+E,M}^{k+1}(\mu) \right\| + \left\| R^{k+1}(\mu) \right\| \right)$$

The residuals $R_{,M}$ measure the empirical interpolation error, e.g.*

$$R_{*,M}^{k+1,\nu} := \sum_{m=M}^{M+M'} l_m^* \left[u_{\text{red}}^{k+1,\nu} \right] \xi_m$$

A posteriori error estimator

Theorem (A posteriori error estimator cont.)

The residuals R^k are *efficiently* computable:

$$\begin{aligned}\Delta t^2 \|R^{k+1}\|^2 &= \langle \Delta t R^{k+1}, \Delta t R^{k+1} \rangle \\ &= (\mathbf{a}^{k+1} - \mathbf{a}^k)^T \mathbf{M} (\mathbf{a}^{k+1} - \mathbf{a}^k)^T \\ &\quad + 2\Delta t \left(\mathbf{l}_I [\mathbf{a}^{k+1}] + \mathbf{l}_E [\mathbf{a}^k] \right)^T \mathbf{C} (\mathbf{a}^{k+1} - \mathbf{a}^k) \\ &\quad + \Delta t^2 \left(\mathbf{l}_I [\mathbf{a}^{k+1}] + \mathbf{l}_E [\mathbf{a}^k] \right)^T \mathbf{X} \left(\mathbf{l}_I [\mathbf{a}^{k+1}] + \mathbf{l}_E [\mathbf{a}^k] \right).\end{aligned}$$

Example: I. Burgers Equation

Burgers Equation

$$\partial_t u - \nabla \mathbf{v} u^{\mu_1} = 0 \quad (2)$$

with (implicit) finite volume discretization with Engquist Osher flux.

Example: I. Burgers Equation

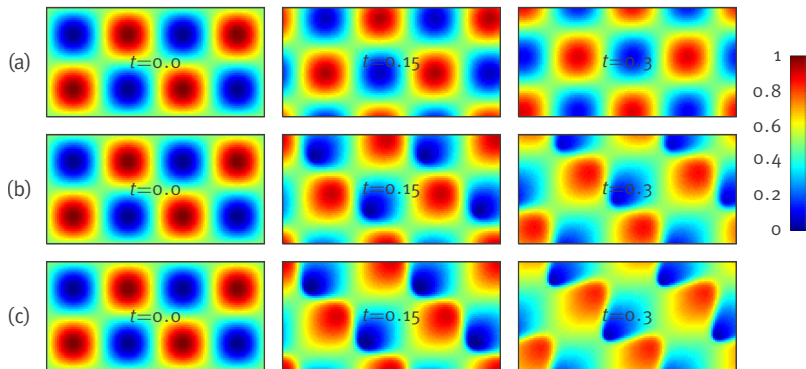
Burgers Equation

$$\partial_t u - \nabla \mathbf{v} u^{\mu_1} = 0 \quad (2)$$

with (implicit) finite volume discretization with Engquist Osher flux.

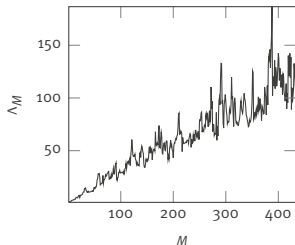
- ▶ Parameter vector $\boldsymbol{\mu} := (\mu_1) \in [0, 2]$.
- ▶ smooth initial data.
- ▶ rectangular 120×60 grid with $K = 100$ time steps.

Example I: Solution snapshots



Example I: Empirical interpolation of $\mathcal{L}_{h,l}$

(a) Lebesgue constant



(b) EI-greedy

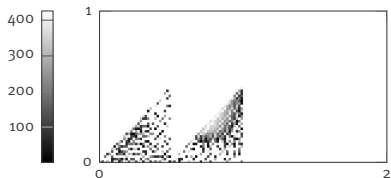


Illustration of interpolation DOF selection for Burgers problem. DOFs corresponding to darker points are selected first.

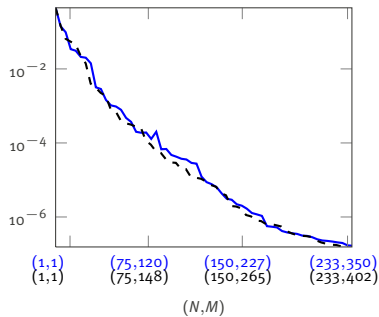
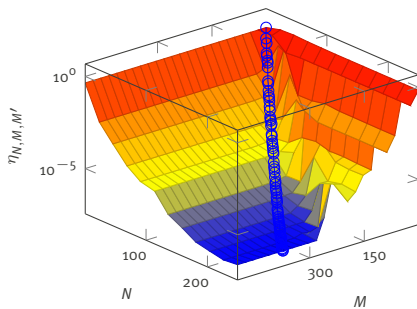
Example I: Table

- ▶ $\dim(\mathcal{W}_h)$ 9600
- ▶ $\nu_{\max} \approx 1 - 20$
- ▶ $\#M_{\text{train}}$ 28

N	M	\emptyset -runtime[s]	max. error	\emptyset -offline time[h]
7,200	0	90.01	0.00	0
42	83	4.42	$1.15 \cdot 10^{-3}$	0.96
83	166	6.23	$6.03 \cdot 10^{-5}$	1.34
125	250	8.99	$7.43 \cdot 10^{-6}$	1.74
166	333	11.6	$8.33 \cdot 10^{-7}$	2.23
208	416	15.64	$2.47 \cdot 10^{-7}$	2.78
249	499	19.56	$2.38 \cdot 10^{-7}$	3.4

N	M	\emptyset -runtime[s]	max. error	\emptyset -offline time[h]
0	-1	90.01	0.00	0
42	72	4.44	$1.73 \cdot 10^{-3}$	0.54
83	144	6.04	$5.74 \cdot 10^{-5}$	1.09
125	216	8.37	$7.30 \cdot 10^{-6}$	1.55
167	288	11.92	$7.63 \cdot 10^{-7}$	2.08
208	360	15.08	$2.31 \cdot 10^{-7}$	2.69
233	402	16.48	$1.55 \cdot 10^{-7}$	3.27

Example I: Error landscape



Example II: Nonlinear Diffusion

Nonlinear Diffusion

Problem definition:

$$\partial_t u - m \Delta u^p = 0 \quad \text{in } \Omega \times [0, 1], \quad u(\cdot, 0) = c_0 + u_0 \quad \text{on } \Omega \times \{0\}$$

Example II: Nonlinear Diffusion

Nonlinear Diffusion

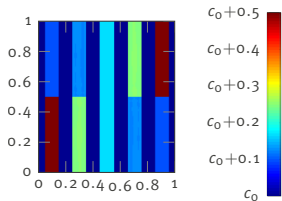
Problem definition:

$$\partial_t u - m \Delta u^p = 0 \quad \text{in } \Omega \times [0, 1], \quad u(\cdot, 0) = c_0 + u_0 \quad \text{on } \Omega \times \{0\}$$

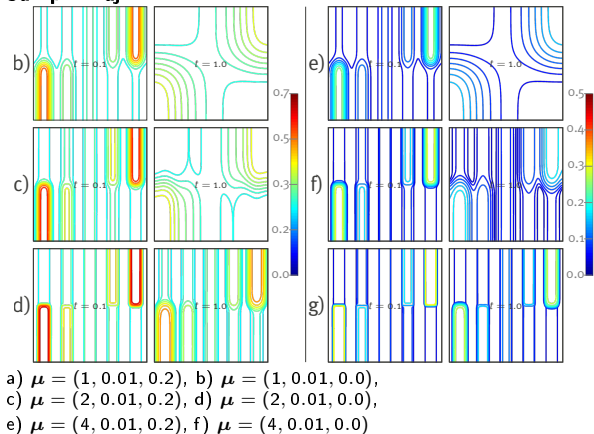
- ▶ Parametrization $\mu = (p, m, c_0) \in [1, 5] \times [0, 0.01] \times [0, 0.2]$
- ▶ $\Omega = [0, 1]^2$ with homogeneous boundary conditions
- ▶ rectangular 100x100 grid with $K = 80$ time steps.

Example II: Solution snapshots

Initial data:

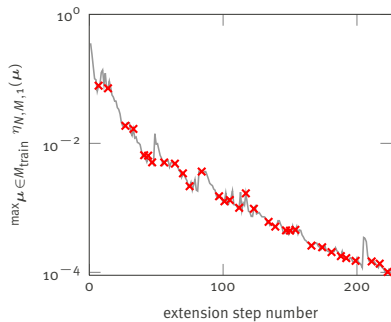


Sample trajectories:

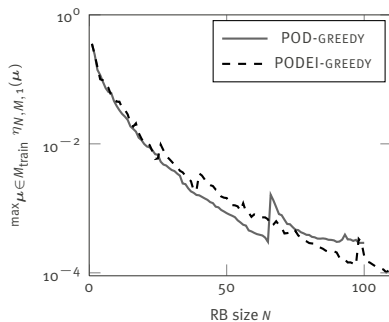


Example II: Greedy error convergence

(a) PODEI-greedy basis discards

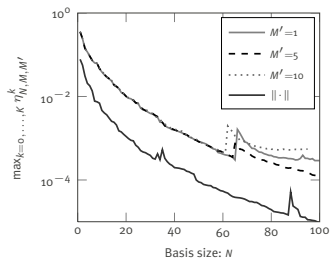


(b) X-greedy error decrease

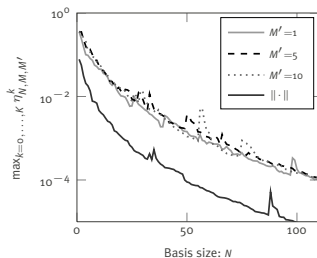


Example II: Greedy error convergence

(a) Estimates for POD-Greedy(M')



(b) Estimates for PODEI-Greedy(M')



Example II: Numerical results

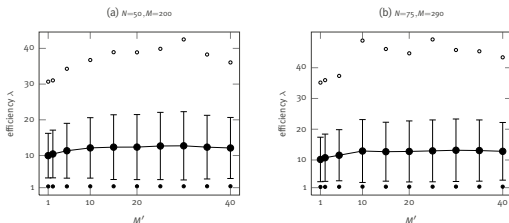
- ▶ $\dim(\mathcal{W}_h)$ 10000
- ▶ $\nu_{\max} \approx 1 - 20$
- ▶ $\#M_{\text{train},0}$ 27.
- ▶ $\#M_{\text{train}}$ 305

N	M	\emptyset -runtime[s]	max. error	\emptyset -offline time[h]
0	0	55.38	0.00	0
17	71	1.57	$3.56 \cdot 10^{-3}$	1.35
33	142	1.95	$8.33 \cdot 10^{-4}$	1.67
50	213	2.51	$2.08 \cdot 10^{-4}$	2.07
66	283	3.19	$5.88 \cdot 10^{-5}$	2.43
83	354	4.07	$5.55 \cdot 10^{-5}$	2.88
99	425	5.3	$4.06 \cdot 10^{-5}$	3.3

N	M	\emptyset -runtime[s]	max. error	\emptyset -offline time[h]
0	-1	55.38	0.00	0
19	72	1.61	$3.01 \cdot 10^{-3}$	0.16
37	143	2.07	$7.90 \cdot 10^{-4}$	0.45
56	215	2.67	$1.66 \cdot 10^{-4}$	1.01
74	286	3.6	$6.36 \cdot 10^{-5}$	1.69
93	358	4.83	$3.54 \cdot 10^{-5}$	2.72
111	429	6.55	$1.96 \cdot 10^{-5}$	4.02

Example II: A posteriori error estimator

Efficiency of error estimator η : $\lambda(\mu) := \frac{\eta(\mu)}{\|u_h(\mu) - u_{\text{red}}(\mu)\|}$



Error bar plot showing mean and standard deviation of error estimator efficiency over a sample of 100 random parameters for different values of M' . The dots indicate the minimum (●) and maximum (○) efficiency.

Other numerical experiments

- ▶ Richards Equation with simple geometry transformation
- ▶ Two-phase flow in porous media (without parametrization and very simple)

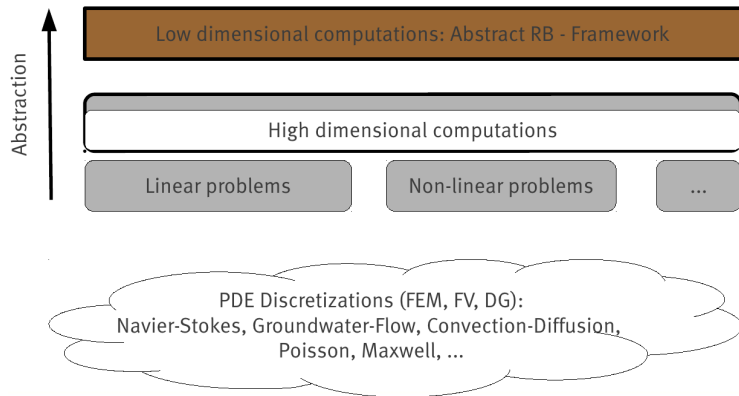
Numerical results

- ▶ Problems are implemented with our software package RBmatlab (<http://www.morepas.org/software>).
- ▶ Computations are executed on compute nodes of the PALMA cluster at the university of Münster with Intel Xeon Westmere X5650, 2,67 GHz processors and 24 GB RAM per node.

Goals specification

- ▶ Abstract reduced basis framework
- ▶ Re-use of (existing) implementations of PDE discretizations
- ▶ Short implementation time for the reduced simulations

Abstract Software concept



- ▶ Control of offline algorithms
 - ▶ POD-greedy, EI-greedy, PODEI-greedy, . . .
 - ▶ Gathering and post-processing of reduced matrices
- ▶ Low-dimensional computations
 - ▶ Reduced simulations
 - ▶ A posteriori error estimation
- ▶ Low-dimensional data visualization
- ▶ (Implemented in Matlab)

High-dimensional computations ()

- ▶ Storage / manipulation of reduced spaces
- ▶ Efficient high dimensional linear algebra algorithms
 - ▶ POD, orthonormalization, Gram-Matrix computations
- ▶ Parametrization
- ▶ (Implemented in C++)
 - ▶ based on DUNE core modules (<http://dune-project.org>)

The glue

- Communication between Dune-RB and RBmatlab can be realized by
 1. compiling Dune-RB example as (mex-) library for matlab, or
 2. TCP/IP communication between two stand-alone applications.



Proof of concept (linear diffusion)

```
RBMATLAB > matlab
```

```
< M A T L A B (R) >  
Copyright 1984-2008 The MathWorks, Inc.  
Version 7.6.0.324 (R2008a)  
February 10, 2008
```

```
To get started, type one of these: helpwin, helpdesk, or demo  
For product information, visit www.mathworks.com.
```

```
starting up rbmatlab in directory:  
/home/martin/projects/rbm-results/rbmatlab  
Using the following directory for large temporary data:  
/tmp  
Using the following directory for data files storing results:  
/home/martin/projects/rbm-results/results  
Using the following directory as RBMATLABHOME:  
/home/martin/projects/rbm-results/rbmatlab  
skipped clearing filecache for function-calls!  
>> █
```

```
DUNE-RB > ./dunerbServer
```

```
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.  
server: waiting for connections...  
█
```

Proof of concept (linear diffusion)

```
< M A T L A B (R) >  
Copyright 1984-2008 The MathWorks, Inc.  
Version 7.6.0.324 (R2008a)  
February 10, 2008
```

To get started, type one of these: helpwin, helpdesk, or demo
.
For product information, visit www.mathworks.com.

```
starting up rbmatlab in directory:  
/home/martin/projects/rbm-results/rbmatlab  
Using the following directory for large temporary data:  
/tmp  
Using the following directory for data files storing results:  
/home/martin/projects/rbm-results/results  
Using the following directory as RBMATLABHOME:  
/home/martin/projects/rbm-results/rbmatlab  
skipped clearing filecache for function-calls!  
>>
```

```
>> [a,b,c] = ...  
mexclient('echo', [1, 2], ...  
          struct('field', [1,2]), ...  
          { [1,2], [3,4] });
```

```
client: connect: Connection refused  
Warning: connection to ::1 failed
```

```
client connected to 127.0.0.1  
copying argument no. 0  
copying argument no. 1  
copying argument no. 2  
>> []
```

```
DUNE-RB > ./dunerbServer  
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.  
server: waiting for connections...  
server: got connection from 127.0.0.1
```

```
Received call for processing 'echo'  
with 4 arguments and 3 return values.  
=====
```

```
copying argument no. 0  
copying argument no. 1  
copying argument no. 2
```

Proof of concept (linear diffusion)

```
/home/martin/projects/rbm-results/rbmatlab  
skipped clearing filecache for function-calls!
```

```
>>  
>> [a,b,c] = ...  
mexclient('echo', [1, 2], ...  
           struct('field', [1,2]), ...  
           { [1,2], [3,4] });
```

```
client: connect: Connection refused
```

```
Warning: connection to ::1 failed
```

```
client connected to 127.0.0.1
```

```
copying argument no. 0
```

```
copying argument no. 1
```

```
copying argument no. 2
```

```
>> a
```

```
a =
```

```
    1    2
```

```
>> b
```

```
b =
```

```
    field: [1 2]
```

```
>> c
```

```
c =
```

```
 [1x2 double]    [1x2 double]
```

```
>> █
```

```
DUNE-RB > ./dunerbServer
```

```
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.
```

```
server: waiting for connections...
```

```
server: got connection from 127.0.0.1
```

```
Received call for processing 'echo'  
with 4 arguments and 3 return values.
```

```
=====
```

```
copying argument no. 0
```

```
copying argument no. 1
```

```
copying argument no. 2
```

```
█
```

Proof of concept (linear diffusion)

```
< M A T L A B (R) >
Copyright 1984-2008 The MathWorks, Inc.
Version 7.6.0.324 (R2008a)
February 10, 2008
```

To get started, type one of these: helpwin, helpdesk, or demo

For product information, visit www.mathworks.com.

```
starting up rbmatlab in directory:
/home/martin/projects/rbm-results/rbmatlab
Using the following directory for large temporary data:
/tmp
Using the following directory for data files storing results:
/home/martin/projects/rbm-results/results
Using the following directory as RBMATLABHOME:
/home/martin/projects/rbm-results/rbmatlab
skipped clearing filecache for function-calls!
>>
```

```
>> % load model parameters
```

```
>>
```

```
>> model = convdiff_dune_model;
```

```
Warning: Name is nonexistent or not a directory: mexclient.
```

```
> In path at 110
```

```
    In addpath at 87
```

```
    In convdiff_dune_model at 95
```

```
client: connect: Connection refused
```

```
Warning: connection to ::1 failed
```

```
client connected to 127.0.0.1
```

```
>> □
```

```
DUNE-RB > ./dunerbServer
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.
server: waiting for connections...
server: got connection from 127.0.0.1

Received call for processing 'init_model'
with 1 arguments and 1 return values.
=====

read discfunclist_xdr from headerfile, size = 20
Using the explicit ode solver! In order to use a different discretization, change the 'DISCRETIZATION' make variable

Received call for processing 'get_mu'
with 1 arguments and 1 return values.
=====

Received call for processing 'rb_symbolic_coefficients'
with 1 arguments and 1 return values.
=====
```

Proof of concept (linear diffusion)

```
> In path at 110
  In addpath at 87
  In convdiff_dune_model at 95
client: connect: Connection refused
Warning: connection to ::1 failed
```

```
client connected to 127.0.0.1
```

```
>>
>> model.rb_problem_type
```

```
ans =
```

```
lin_evol
```

```
>> model.RB_generation_mode
```

```
ans =
```

```
greedy_uniform_fixed
```

```
>> model.RB_stop_Nmax
```

```
ans =
```

```
20
```

```
>> model.T % this is read from DUNE-RB <<<<<<
```

```
ans =
```

```
1
```

```
>> █
```

```
DUNE-RB > ./dunerbServer
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.
server: waiting for connections...
server: got connection from 127.0.0.1
```

```
Received call for processing 'init_model'
with 1 arguments and 1 return values.
=====
```

```
read discfunclist_xdr from headerfile, size = 20
Using the explicit ode solver! In order to use a different discretization, change the 'DISCRETIZATION' make variable
```

```
Received call for processing 'get_mu'
with 1 arguments and 1 return values.
=====
```

```
Received call for processing 'rb_symbolic_coefficients'
with 1 arguments and 1 return values.
=====
```

```
█
```

Proof of concept (linear diffusion)

```
ans =  
lin_evol  
>> model.RB_generation_mode  
ans =  
greedy_uniform_fixed  
>> model.RB_stop_Nmax  
ans =  
20  
>> model.T % this is read from DUNE-RB <<<<<<  
ans =  
1  
>>  
>>  
>>  
>> % generate high dimensional model specific data, like e.g.  
>> % the grid  
>>  
>> model_data = gen_model_data(model);  
>> █
```

```
DUNE-RB > ./dunerbServer  
YaspGridParameterBlock: Parameter 'overlap' not specified, defaulting to '0'.  
server: waiting for connections...  
server: got connection from 127.0.0.1  
  
Received call for processing 'init_model'  
with 1 arguments and 1 return values.  
=====
```

read discfunclist_xdr from headerfile, size = 20
Using the explicit ode solver! In order to use a different discretization, change the 'DISCRETIZATION' make variable

```
Received call for processing 'get_mu'  
with 1 arguments and 1 return values.  
=====
```

Received call for processing 'rb_symbolic_coefficients'
with 1 arguments and 1 return values.
=====

```
Received call for processing 'gen_model_data'  
with 1 arguments and 1 return values.  
=====
```

█

Proof of concept (linear diffusion)

[illegible]

```
Received call for processing 'rb_symbolic_coefficients'
with 1 arguments and 1 return values.
```

```
Received call for processing 'gen_model_data'
with 1 arguments and 1 return values.
```

```
Received call for processing 'set_mu'
with 2 arguments and 0 return values.
```

```
Received call for processing 'detailed_simulation'
with 1 arguments and 1 return values.
```

```
opening file: ./grape//solution.series
```

```
Received call for processing 'set_mu'
with 2 arguments and 0 return values.
```

```
Received call for processing 'detailed_simulation'
with 1 arguments and 1 return values.
```

```
opening file: ./grape//solution.series
```

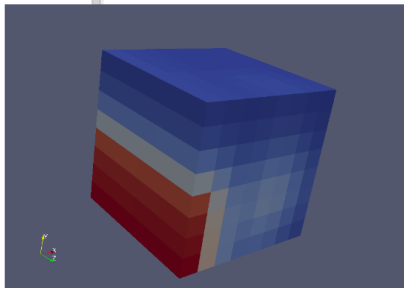
Proof of concept (linear diffusion)

```
>> % Just for fun: Do a DETAILED simulation in DUNE-RB >>>>>
>>
>> % first set the parameter mu
>> model = set_mu(model, [0.0 0.5 1.0]);
>>
>> % then run the simulation
>> tic; sim_data = detailed_simulation(model, model_data); toc
Elapsed time is 11.795319 seconds.
>>
```

11.8 seconds!

```
Received call for processing 'rb_symbolic_coefficients'
with 1 arguments and 1 return values.
=====
```

```
Received call for processing 'gen_model_data'
with 1 arguments and 1 return values.
=====
```



opening file: ./grape//solution.series

Proof of concept (linear diffusion)

```
>> % Generate the reduced basis with the POD-Greedy algorithm
>> % in DUNE-RB >>>>>>>>
>>
>> detailed_data = gen_detailed_data(model, model_data);
Starting RB extension loop
```

```
Detected maximum error prediction 0.044006 for mu=[0.001
1 0.5]
Extended RB to length 2
```

```
Detected maximum error prediction 0.015456 for mu=[0 1
Extended RB to length 3
```

```
Detected maximum error prediction 0.012877 for mu=[0
1 0.5]
Extended RB to length 4
```

```
Detected maximum error prediction 0.01064 for mu=[0
0.5]
Extended RB to length 5
```

```
Detected maximum error prediction 0.0084073 for mu=[0.001
0.5 1]
Extended RB to length 6
```

```
Detected maximum error prediction 0.0073233 for mu=[0 1 1]
Extended RB to length 7
```

```
Detected maximum error prediction 0.0055443 for mu=[0 1 1]
Extended RB to length 8
```

```
Detected maximum error prediction 0.0048443 for mu=[0
1 0.5]
```

```
Received call for processing 'rb_operators'
with 2 arguments and 1 return values.
=====
```

```
Received call for processing 'get_mu'
with 1 arguments and 1 return values.
=====
```

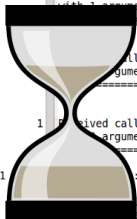
```
Received call for processing 'set_mu'
with 1 arguments and 0 return values.
=====
```

```
1 Received call for processing 'rb_extension_PCA'
with 1 arguments and 1 return values.
=====
```

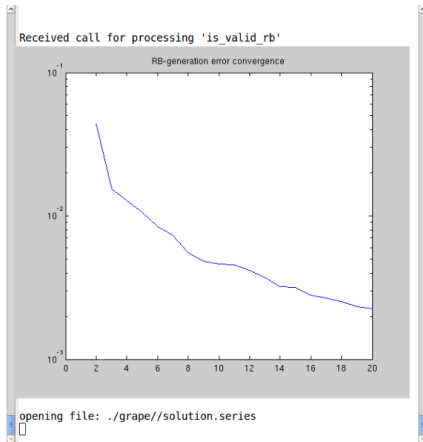
```
./grape//solution.series
```

```
Received call for processing 'set_mu'
with 1 arguments and 0 return values.
=====
```

```
Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.
=====
```



Proof of concept (linear diffusion)

[illegible]

Proof of concept (linear diffusion)

```
>> % The matrices are all of small sizes: e.g. the explicit
>> % discretization operator: LL_E
>>
>> sizes=cellfun(@(X) size(X), reduced_data.LL_E, ...
    'UniformOutput', false);
>> sizes{:}

ans =

    20    20

ans =

    20    20

ans =

    20    20

ans =

    20    20

ans =

    20    20

>>
>> █
```

```
Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.
=====
```

```
Received call for processing 'rb_operators'
with 2 arguments and 1 return values.
=====
```

```
Received call for processing 'set_mu'
with 2 arguments and 0 return values.
=====
```

```
Received call for processing 'reconstruct_and_compare'
with 2 arguments and 0 return values.
=====
```

```
opening file: ./grape//solution.series
```

```
Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.
=====
```

```
Received call for processing 'rb_operators'
with 2 arguments and 1 return values.
=====
```

Proof of concept (linear diffusion)

```
>> % Now fast reduced simulations are possible in RBMATLAB
>> % without any communication to DUNE-RB
>> model = model.set_mu(model, [0 0.5 1], true);
>> tic; rb_sim_data=rb_simulation(model, reduced_data); toc
Elapsed time is 0.020223 seconds.
```

```
>>
>> rb_sim_data
```

```
rb_sim_data =
    a: [20x113 double]
   Delta: [1x113 double]
   LL_I: [20x20 double]
   LL_E: [20x20 double]
```

```
>>
>> % Error estimator at end time:
>>
```

```
>> rb_sim_data.Delta(end)
```

```
ans =
    0.0019
```

```
>>
>>
>>
>>
>> █
```

0.02 seconds!

Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.

Received call for processing 'rb_operators'
with 2 arguments and 1 return values.

Received call for processing 'set_mu'
with 2 arguments and 0 return values.

Received call for processing 'reconstruct_and_compare'
with 2 arguments and 0 return values.

opening file: ./grape//solution.series

Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.

Received call for processing 'rb_operators'
with 2 arguments and 1 return values.

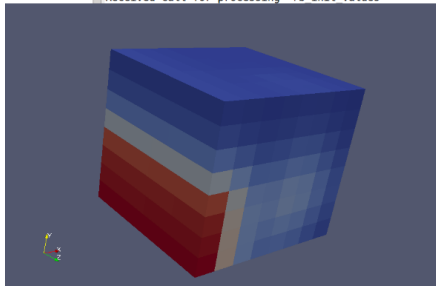
Proof of concept (linear diffusion)

```
>> % Of course, the solution can be reconstructed in
>> % DUNE-RB >>>>>>>>
>>
>> tic;...
dummy=rb_reconstruction(model, detailed_data, rb_sim_data); toc
Elapsed time is 0.359108 seconds.
```

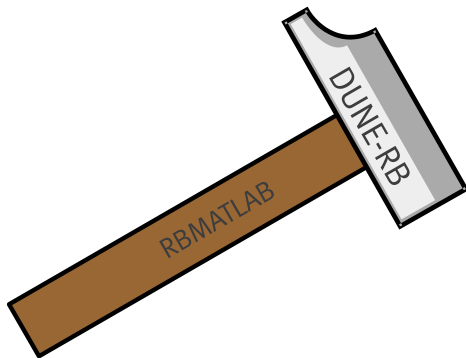
```
Received call for processing 'reconstruct_and_compare'
with 2 arguments and 0 return values.
```

```
opening file: ./grape//solution.series
```

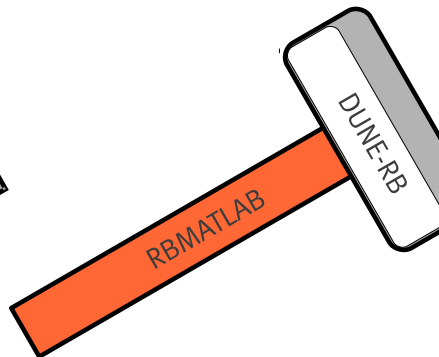
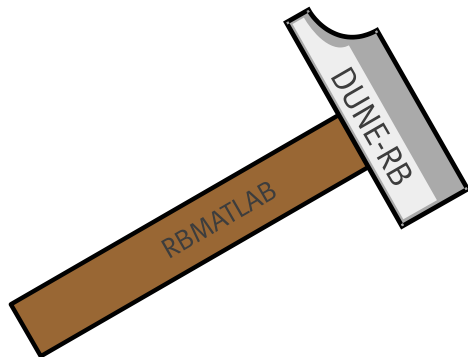
```
Received call for processing 'rb init values'
```



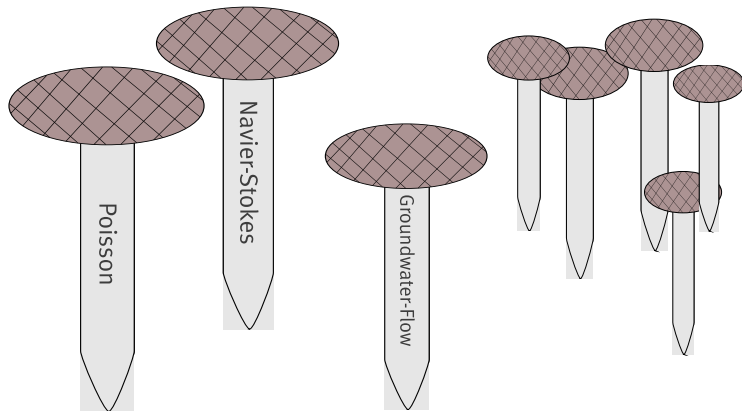
So, we have a hammer for linear problems...



...and also one for nonlinear problems.



But how do we make our problems look like nails?



Interface to (linear) PDE discretizations

```
>> % Generate the reduced basis with the POD-Greedy algorithm
>> % in DUNE-RB >>>>>>>
>> detailed_data = gen_detailed_data(model, model_data);
Starting RB extension loop
```

```
Detected maximum error prediction 0.044006 for mu=[0.001
1 0.5]
Extended RB to length 2
```

```
Detected maximum error prediction 0.015456 for mu=[0 1 1]
Extended RB to length 3
```

```
Detected maximum error prediction 0.012877 for mu=[0
1 0.5]
Extended RB to length 4
```

```
Detected maximum error prediction 0.01064 for mu=[0
0.5]
Extended RB to length 5
```

```
Detected maximum error prediction 0.0084073 for mu=[0.001
0.5 1]
Extended RB to length 6
```

```
Detected maximum error prediction 0.0073233 for mu=[0 1 1]
Extended RB to length 7
```

```
Detected maximum error prediction 0.0055443 for mu=[0 1 1]
Extended RB to length 8
```

```
Detected maximum error prediction 0.0048443 for mu=[0
1 0.5]
```

```
Received call for processing 'rb_operators'
with 2 arguments and 1 return values.
=====
```

```
Received call for processing 'get_mu'
with 1 arguments and 1 return values.
=====
```

```
Received call for processing 'set_mu'
with 2 arguments and 0 return values.
=====
```

```
1 Received call for processing 'rb_extension_PCA'
with 3 arguments and 1 return values.
=====
```

```
opening file: ./grape//solution.series
```

```
Received call for processing 'set_mu'
with 2 arguments and 0 return values.
=====
```

```
Received call for processing 'rb_init_values'
with 2 arguments and 1 return values.
=====
```

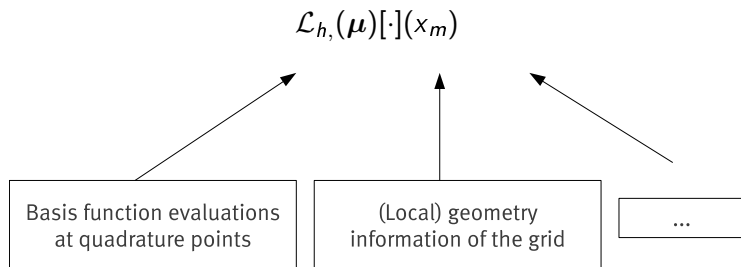
Interface to (linear) PDE discretizations

Return affinely decomposed operator parts:

- ▶ components: \mathcal{L}_h^q , and
- ▶ coefficients: $\sigma^q(\mu)$

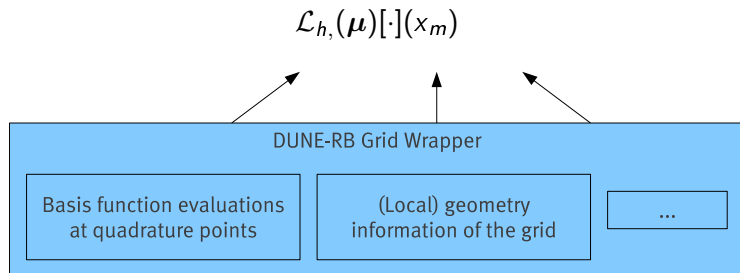
Interface to (non-linear) PDE discretizations

Common dependencies for local operator evaluations



Interface to (non-linear) PDE discretizations

Common dependencies for local operator evaluations



Dune-RB grid wrapper

- ▶ During detailed simulation
 - ▶ **Delegate** calls **directly** to the grid
- ▶ During offline phase
 - ▶ **Store** all grid and function space information on the **subgrid** in **low-dimensional** data structures
- ▶ During online phase
 - ▶ **Delegate** calls **low-dimensional** data structures generated in offline phase.

More information

<http://morepas.org/software>

Conclusion

- ▶ Model order reduction of general (scalar) parametrized evolution schemes
- ▶ with reduced basis methods and empirical interpolation for discrete operators
- ▶ Rigorous error control via a posteriori error estimator is possible.

Conclusion

- ▶ Model order reduction of general (scalar) parametrized evolution schemes
- ▶ with reduced basis methods and empirical interpolation for discrete operators
- ▶ Rigorous error control via a posteriori error estimator is possible.

Future work

- ▶ Dealing with steep gradients in solution snapshots (non-linear reduced bases?)
- ▶ Variable time step width
- ▶ 2-Phase flow system
- ▶ Improve software